Network Controller Emulation on a Sidecore for Unmodified Virtual Machines

TECHNION
Israel Institute
of Technology
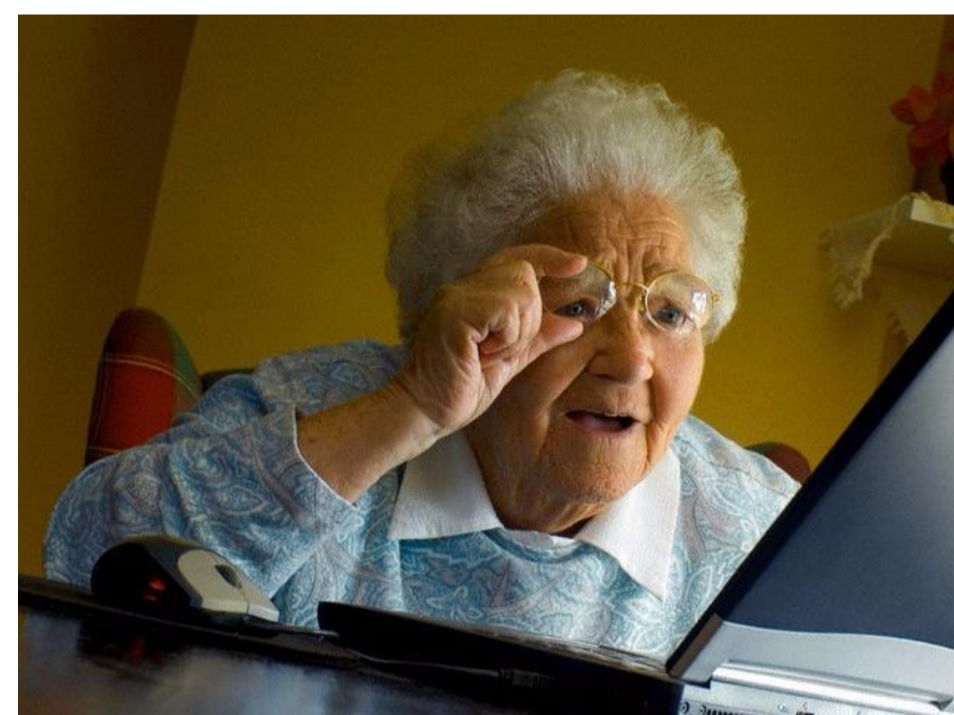
Computer Science
Department

Arthur Kiyanovski
Dan Tsafrir

Virtual Machine Send Emulation

GUEST

Virtual Memory

Driver

Packets

Regs

1) WRITE
2) NOTIFY
3) EXIT
4) READ
6) RESUME

Unmapped!

HOST

Hypervisor

5) Emulate and Send packet to real NIC

Driver

Virtual Memory

Packets

Control Registers

5.1) WRITE
5.2) NOTIFY
5.3) READ PACKET
5.4) SEND PACKET
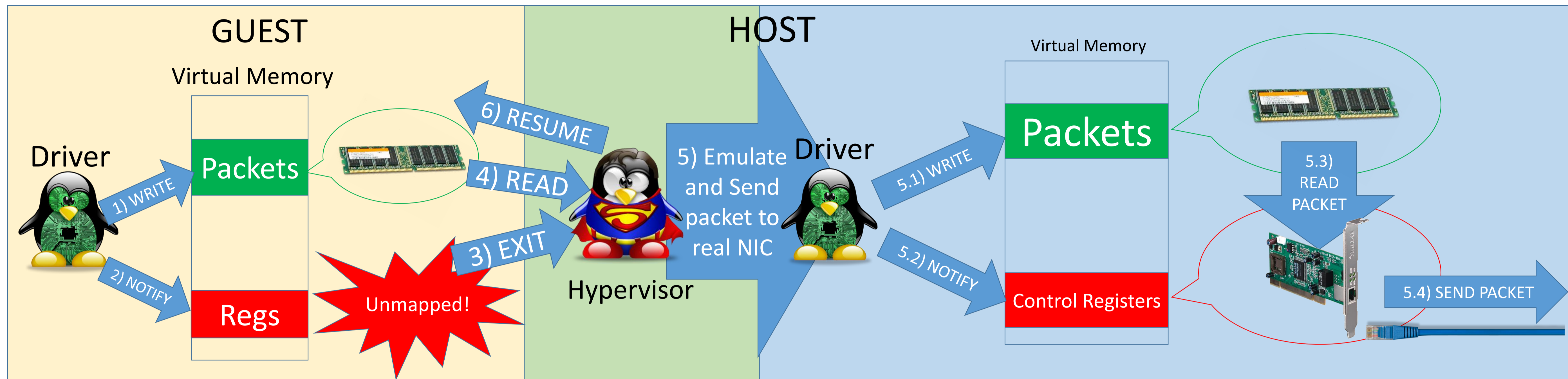
Emulation VS Paravirtualization

Emulation (E1000)
- HOST
- GUEST
- Single packet send Emulation
- Many control related exits each packet
- Single Packet preparation

Paravirtualization (Virtio)
- HOST
- GUEST
- Single exit per batch
- Many packets Emulated at once (batching)
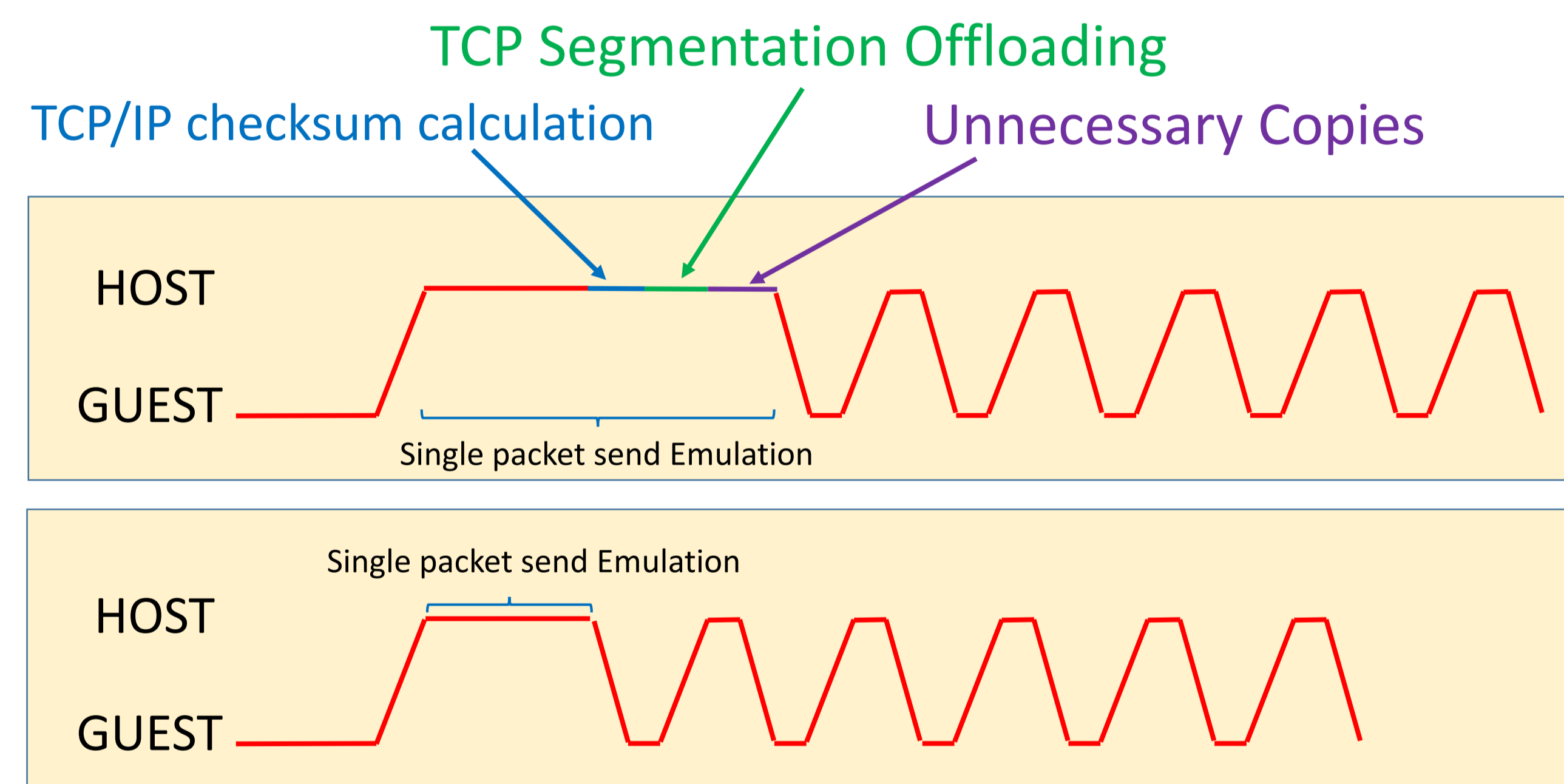- Many packets prepared at once (batching)
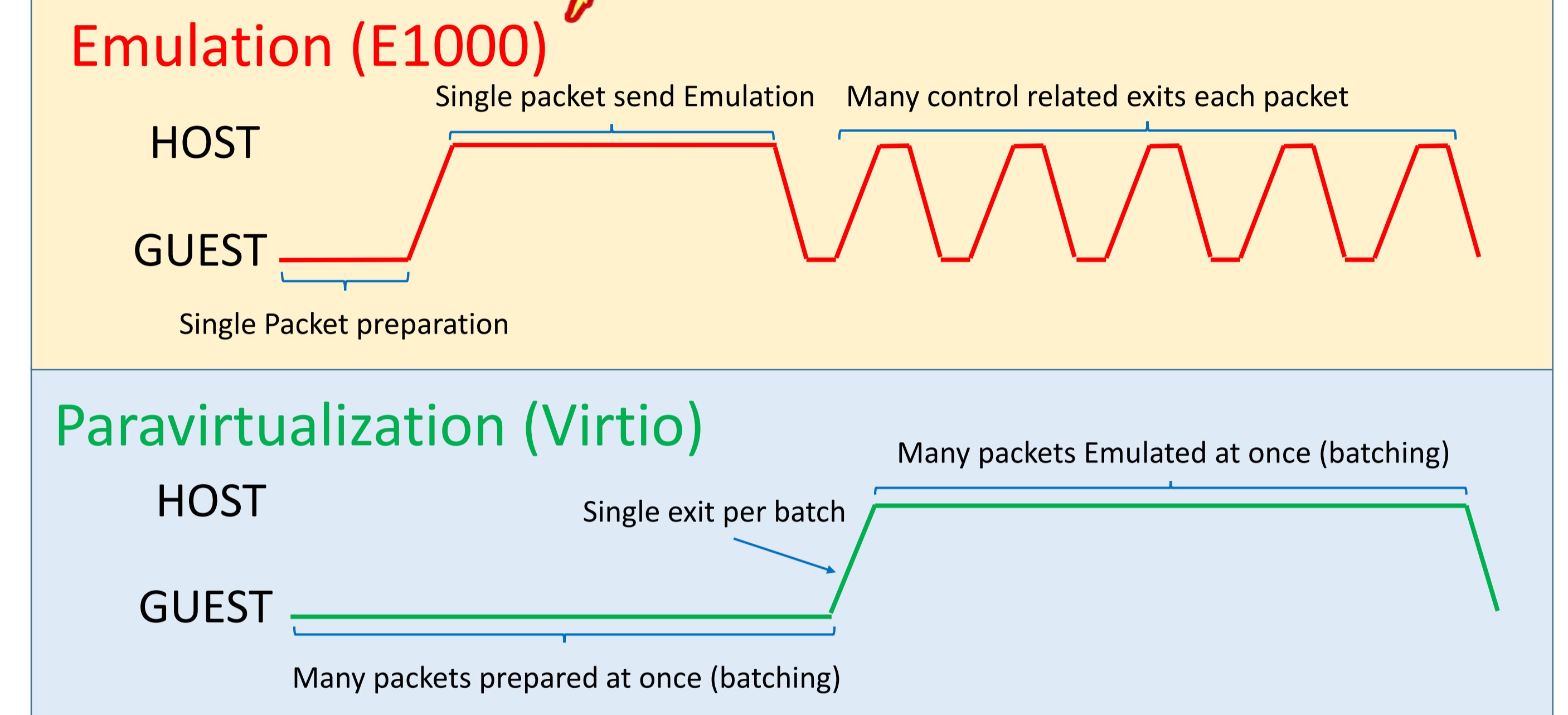
Why Emulation?

Client doesn't need to install anything!
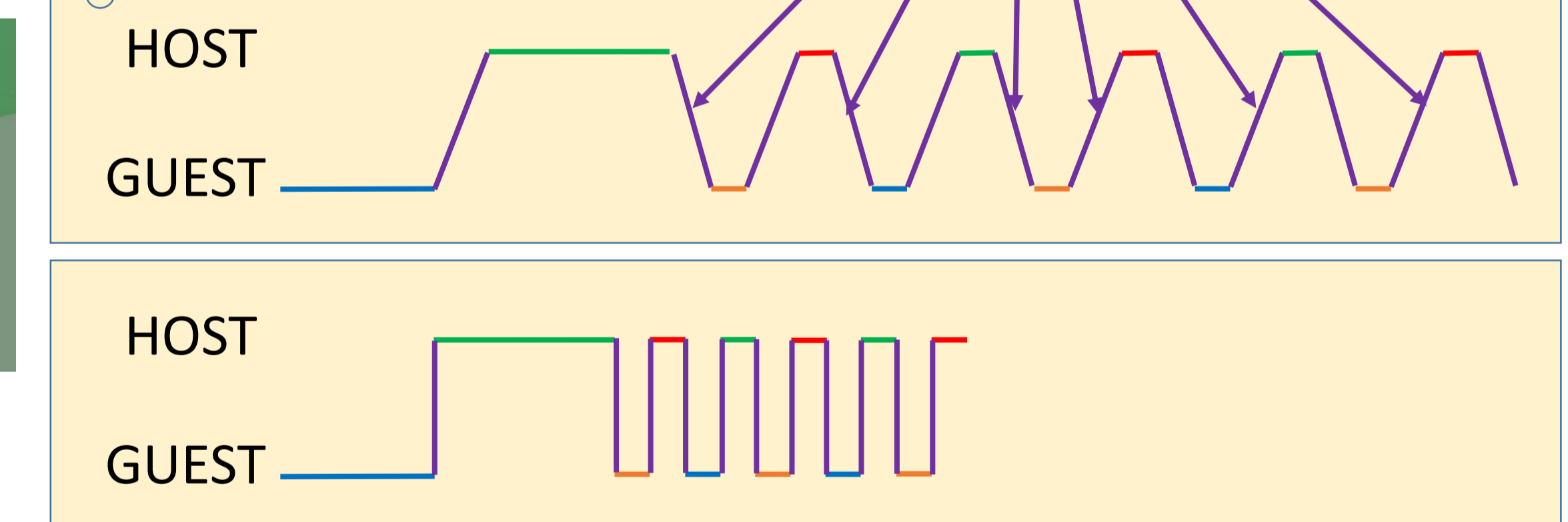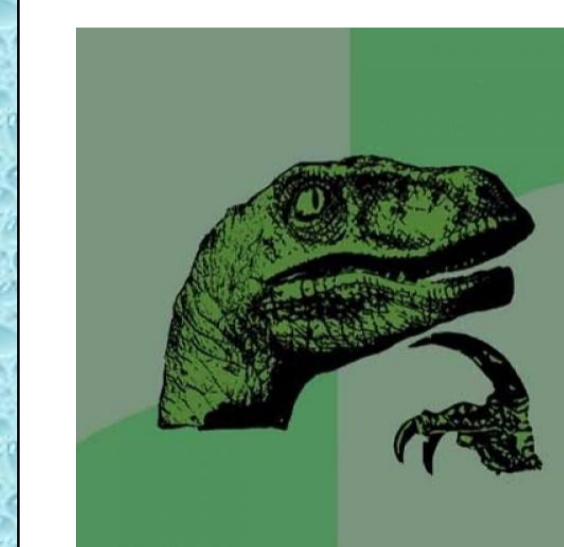
Vendor doesn't need to develop drivers for every OS!

Improving Emulation 1

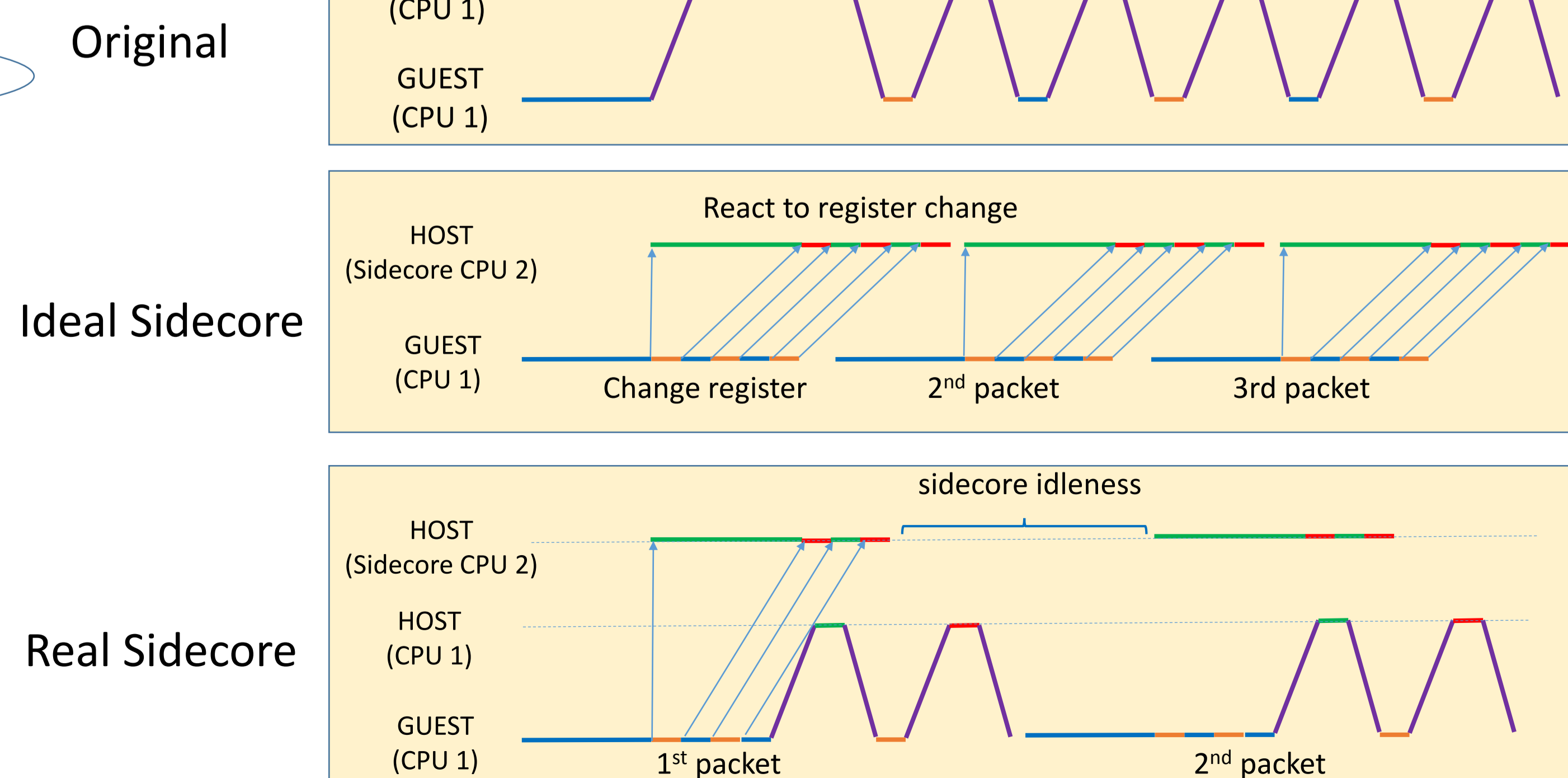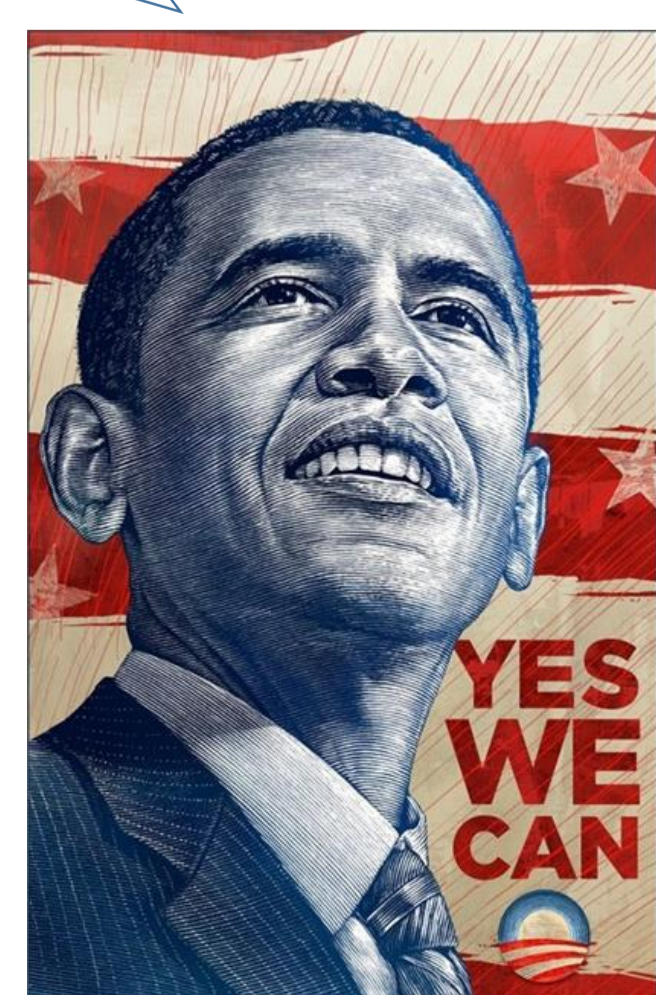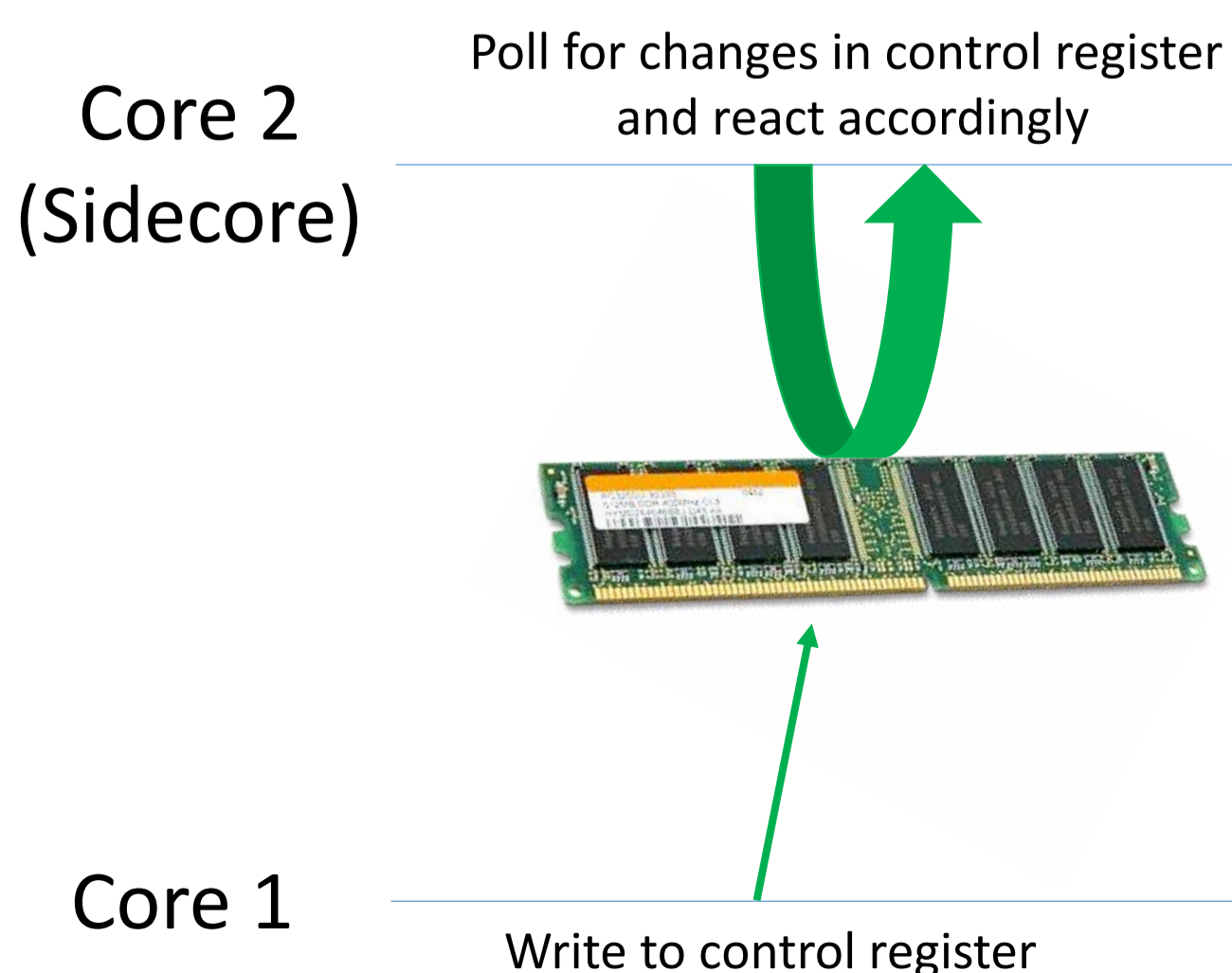TCP Segmentation Offloading
TCP/IP checksum calculation
Unnecessary Copies

HOST
GUEST
Single packet send Emulation

HOST
GUEST
Single packet send Emulation

Improving Emulation 2?

If only we could remove the Exit overhead...

HOST
GUEST

HOST
GUEST

Improving Emulation 2!

We can remove exits using a Sidecore!!

Core 2 (Sidecore)

Poll for changes in control register and react accordingly

Core 1

Write to control register

Original
Ideal Sidecore
Real Sidecore

HOST (CPU 1)
GUEST (CPU 1)

HOST (Sidecore CPU 2)
GUEST (CPU 1)
React to register change
Change register
2nd packet
3rd packet

HOST (Sidecore CPU 2)
HOST (CPU 1)
GUEST (CPU 1)
sidecore idleness
1st packet
2nd packet

Current Results

Baseline E1000 emulation
500 Mbps
Improved E1000 emulation
4000 Mbps (X8)
Partial Sidecore E1000 emulation
8000 Mbps (X16)

VS

Baseline Virtio Paravitualization
8000 Mbps

Future Work

Improve Sidecore implementation to be as close to Ideal as possible.

Implement E1000 Emulation in kernel space (like Virtio Vhost)

Use single sidecore for multiple guests (like ELVIS)